What is claimed is:

1. A method for controlling a network system, comprising:

(a) defining a first event that occurs in said network system, wherein said first event is defined via software and corresponds to information transmitted over said network system in at least one of application, presentation, session, transport and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said first event.

2. The method as claimed in claim 1, wherein said first event is defined in at least the network layer of said communication model.

3. The method as claimed in claim 1, wherein said first event is defined in at least the transport layer of said communication model.

4. The method as claimed in claim 1, wherein said first event is defined in at least the session layer of said communication model.

5. The method as claimed in claim 1, wherein said first event is defined in at least the presentation layer of said communication model.

6. The method as claimed in claim 1, wherein said first event is defined in at least the application layer of said communication model.

7. The method as claimed in claim 1, further comprising:

(c) defining a second event via said software, wherein said second event is defined at least partially with said first event.

8. The method as claimed in claim 1, wherein said operation (a) comprises:

(a1) defining a second event via said software; and

(a2) defining said first event at least partially with said second event.

9. The method as claimed in claim 1, wherein said operation (a) comprises:

(a1) defining a matching operation via said software, wherein said matching operation detects an occurrence of a variable in information transmitted over said network system, wherein said variable is identified in said matching operation; and

(a2) defining said first event at least partially with said matching operation.

10. The method as claimed in claim 9, wherein said matching operation identifies whether or not a series of data within a process flow corresponds to said variable.

11. The method as claimed in claim 9, wherein said matching operation identifies whether or not an occurrence of a predefined event corresponds to said variable, and

wherein said matching operation determines that said predefined event has occurred when said predefined event corresponds to said variable.

12. The method as claimed in claim 9, further comprising:

(c) repeatedly performing said matching operation within a process flow.

13. The method as claimed in claim 12, further comprising:

(d) discontinuing performance of said matching operation when said matching operation detects said occurrence of said variable.

14. The method as claimed in claim 12, wherein said operation (c) comprises:

(c1) updating whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said matching operation detects said occurrence of said variable, and

wherein said matching operation is said failure when said matching operation does not detect said occurrence of said variable.

15. The method as claimed in claim 9, wherein said matching operation is performed on a plurality of process flows.

16. The method as claimed in claim 9, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence of said variable in said information is detected after said specified location.

17. The method as claimed in claim 9, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence of said variable in

said information is detected at said specified location.

18. The method as claimed in claim 9, wherein said operation (a) further comprises:

(a3) defining an otherwise operation, wherein said otherwise operation is performed when said matching operation is a failure; and

(a4) defining said first event at least partially with said otherwise operation.

19. The method as claimed in claim 18, wherein said operation (a3) comprises:

(a3a) defining a throw operation,

wherein said throw operation identifies resync operation corresponding to said throw operation, and

wherein said throw operation causes a specific operation corresponding to said resync operation to be performed.

20. The method as claimed in claim 18, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

21. The method as claimed in claim 1, wherein said operation (a) comprises:

(a1) defining a concurrent operation, wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed; and

(a2) defining said first event at least partially with said concurrent operation.

22. The method as claimed in claim 21, wherein said concurrent operation ceases performing all unsuccessful operations of said first group of operations when one operation of said first group of operations is a success.

23. The method as claimed in claim 22, wherein said operation (a) comprises:

(a3) defining a third operation that is performed if all of said first group of operations are not a success; and

(a4) defining said first event at least partially with said third operation.

24. The method as claimed in claim 9, wherein the matching operation is defined with a modifier that identifies a specific location in said information where said occurrence of said variable is to be detected.

25. The method as claimed in claim 24, wherein said modifier identifies one of (1) a starting offset of an Internet protocol header in a current datagram transmitted on said network system, (2) a starting offset of a payload in a current datagram transmitted on said network system, (3) a starting offset of a transport header in a current datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a current process flow transmitted on said network system.

26. The method as claimed in claim 1, further comprising:

(c) defining a timer via said software, wherein said timer counts time.

27. The method as claimed in claim 26, wherein said timer is capable counting time in a selected time unit,

wherein said selected time unit is selected from one of a plurality of available predefined time units, and

wherein said timer is defined to count time in said selected time unit.

28. The method as claimed in claim 27, wherein one of said predefined time units comprises one of milliseconds and seconds.

29. The method as claimed in claim 26, wherein said timer decrements a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said timer ceases counting and generates an indication that said current count value equals said predefined count value.

30. The method as claimed in claim 26, wherein said timer increments a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said timer ceases counting and generates an indication that said current count value equals said predefined count value.

31. The method as claimed in claim 26, wherein said timer is capable counting time in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said timer is incremented and a second direction in which said count value is decremented, and

wherein said timer is defined to count time in said selected direction.

32. The method as claimed in claim 31, wherein, when said count value equals a predefined count value, said timer ceases counting and generates an indication that said count value equals said predefined count value.

33. The method as claimed in claim 26, further comprising:

(d) providing a timer start operation that instructs said timer to start counting.

34. The method as claimed in claim 26, further comprising:

(d) providing a timer stop operation that instructs said timer to stop counting.

35. The method as claimed in claim 26, further comprising:

(d) providing a timer pause operation that instructs said timer to suspend counting.

36. The method as claimed in claim 35, further comprising:

(e) providing a timer resume operation that instructs said timer to resume counting after said timer has suspended counting.

37. The method as claimed in claim 1, further comprising:

(c) defining a meter via said software, wherein said meter counts quantities of a selected quantity unit.

38. The method as claimed in claim 37, wherein said selected quantity unit is selected from one of a plurality of available predefined quantity units.

39. The method as claimed in claim 38, wherein one of said predefined quantity units comprises one of bits of data, bytes of data, and number of packets.

40. The method as claimed in claim 38, wherein one of said predefined quantity units comprises numbers of events.

41. The method as claimed in claim 38, wherein said one of said predefined quantity units comprises numbers of process flows.

42. The method as claimed in claim 37, wherein said meter counts said quantities of said selected quantity unit in a plurality of process flows.

43. The method as claimed in claim 37, wherein said meter decrements a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said meter ceases counting and generates an indication that said current count value equals said predefined count value.

44. The method as claimed in claim 37, wherein said meter increments a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said meter ceases counting and generates an indication that said current count value equals said predefined count value.

45. The method as claimed in claim 37, wherein said meter is capable counting said quantities in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said meter is incremented and a second direction in which said count value is decremented, and

wherein said meter is defined to count said quantities in said selected direction.

46. The method as claimed in claim 45, wherein, when said count value equals a predefined count value, said meter ceases counting and generates an indication that said count value equals said predefined count value.

47. The method as claimed in claim 37, further comprising:

(d) providing a meter start operation that instructs said meter to start counting.

48. The method as claimed in claim 37, further comprising:

(d) providing a meter stop operation that instructs said meter to stop counting.

49. The method as claimed in claim 37, further comprising:

(d) providing a meter pause operation that instructs said meter to suspend counting.

50. The method as claimed in claim 49, further comprising:

(e) providing a meter resume operation that instructs said meter to resume counting after said meter has suspended counting.

51. The method as claimed in claim 1, wherein said operation (a) comprises:

(a1) defining a mergeFlow operation which merges a current process flow to at least one existing process flow and which allocates joint resources of a computer system for said current process flow and said at least one existing process flow; and

(a2) defining said first event at least partially with said mergeFlow operation.

52. The method as claimed in claim 51, wherein said mergeFlow operation assigns a unique flow index number to said current process flow.

53. The method as claimed in claim 52, wherein said flow index number of said current process flow is different than a flow index number of said at least one existing process flow.

54. The method as claimed in claim 9, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed when said match operation is a success, and

wherein said first event completes execution when said dedicated instruction is executed.

55. The method as claimed in claim 54, wherein the dedicated instruction indicates a successful completion of said first event.

56. The method as claimed in claim 9, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed when said match operation is unsuccessful, and

wherein said first event completes execution when said dedicated instruction is executed.

57. The method as claimed in claim 56, wherein the dedicated instruction indicates an unsuccessful completion of said first event.

58. The method as claimed in claim 55, wherein said first event generates successful completion data when said first event is successfully completed, and

wherein at least a second event utilizes said successful completion data to determine whether or not said first event has been successfully completed.

59. The method as claimed in claim 57, wherein said first event generates unsuccessful completion data when said first event is not successfully completed, and

wherein at least a second event utilizes said unsuccessful completion data to determine whether or not said first event has been successfully completed.

60. The method as claimed in claim 1, wherein said operation (a) comprises:

(a1) defining a variable as part of said first event; and

(a2) defining an alias for said variable as part of said first event, wherein said variable is capable of being referenced via said alias.

61. The method as claimed in claim 60, further comprising:

(c) defining a second event, wherein a variable reference to said alias is part of

said second event and wherein said second event can utilize said variable via said variable reference.

62. The method as claimed in claim 60, wherein said variable has a plurality of subparts, and

wherein said alias is defined as a subset of said variable containing less than all of said subparts such that said variable reference corresponds to only said subset.

63. The method as claimed in claim 62, wherein said subparts are subfields of said variable.

64. The method as claimed in claim 1,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable equals said at least said particular value.

65. The method as claimed in claim 64, wherein said particular operation is a match operation.

66. The method as claimed in claim 65, wherein said particular operation is an

event definition.

67. The method as claimed in claim 1,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable does not equal said at least said particular value.

68. The method as claimed in claim 67, wherein said particular operation is a match operation.

69. The method as claimed in claim 67, wherein said particular operation is an event definition.

70. The method as claimed in claim 1,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable is within said particular range.

71. The method as claimed in claim 70, wherein said particular operation is a match operation.

72. The method as claimed in claim 70, wherein said particular operation is an event definition.

73. The method as claimed in claim 1,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable is not within said particular range.

74. The method as claimed in claim 73, wherein said particular operation is a match operation.

75. The method as claimed in claim 73, wherein said particular operation is an event definition.

76. The method as claimed in claim 61, wherein said alias is defined as a particular subfield of said variable and wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable reference equals said particular value.

77. The method as claimed in claim 76, wherein said particular operation is a match operation.

78. The method as claimed in claim 76, wherein said particular operation is an event definition.

79. The method as claimed in claim 61, wherein said alias is defined as a particular subfield of said variable and wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable reference does not equal said particular value.

80. The method as claimed in claim 79, wherein said particular operation is a match operation.

81. The method as claimed in claim 79, wherein said particular operation is an event definition.

82. The method as claimed in claim 61, wherein said alias is defined as a particular subfield of said variable and wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable reference is within said particular range.

83. The method as claimed in claim 82, wherein said particular operation is a match operation.

84. The method as claimed in claim 82, wherein said particular operation is an event definition.

85. The method as claimed in claim 61, wherein said alias is defined as a particular subfield of said variable and wherein said method further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable

reference is not within said particular range.

86. The method as claimed in claim 85, wherein said particular operation is a match operation.

87. The method as claimed in claim 85, wherein said particular operation is an event definition.

88. The method as claimed in claim 1, wherein said first event is described in at least a first layer of said communication model.

89. The method as claimed in claim 88, wherein said first layer is one of said application layer, said presentation layer, said session layer, said transport layer, and said network layer of said communication model.

90. The method as claimed in claim 89, further comprising:

(c) transforming said first event into actions in said first layer of said communication model.

91. The method as claimed in claim 89, further comprising:

(c) transforming said first event into actions in a second layer of said communication model, wherein said second layer is lower in said communication model than said first layer.

92. The method as claimed in claim 1, wherein said operation (a) comprises:

(a1) defining a first throw operation, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction; and

(a2) defining said first event at least partially with said first throw instruction, and

wherein said method further comprises:

(d) defining a resynchronization operation.

93. The method as claimed in claim 92, further comprising:

(e) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed.

94. The method as claimed in claim 93, wherein said resynchronization operation identifies said first exception name.

95. The method as claimed in claim 93, further comprising:

(f) defining a second throw operation, wherein said second throw operation identifies a second exception name corresponding to said second throw instruction, and

wherein said resynchronization operation identifies said first exception name and said second exception name, and

wherein said operation (e) comprises:

(e1) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

(e2) resynchronizing to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said

second exception name.

96. A method for controlling a network system, comprising:

(a) defining a matching operation that occurs in said network system, wherein said matching operation is defined via software and detects an occurrence corresponding to information transmitted over said network system in at least one of application, presentation, session, transport and network layers of a communication model, and

wherein said occurrence is identified in said matching operation; and

(b) controlling at least a portion of said network system based on said matching operation.

97. The method as claimed in claim 96, wherein said operation (a) comprises:

(a1) defining a first event via said software; and

(a2) defining said first event at least partially with said matching operation.

98. The method as claimed in claim 96, wherein said matching operation identifies whether or not a series of data within a process flow corresponds to said occurrence.

99. The method as claimed in claim 96, wherein said occurrence corresponds to a predefined event, and

wherein said matching operation identifies whether or not said predefined event has occurred.

100. The method as claimed in claim 96, further comprising:

(c) repeatedly performing said matching operation within a process flow.

101. The method as claimed in claim 100, further comprising:

(d) discontinuing performance of said matching operation when said matching operation detects said occurrence.

102. The method as claimed in claim 100, wherein said operation (c) comprises:

(c1) updating whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said matching operation detects said occurrence, and

wherein said matching operation is said failure when said matching operation does not detect said occurrence.

103. The method as claimed in claim 96, wherein said matching operation is performed on a plurality of process flows.

104. The method as claimed in claim 96, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected after said specified location.

105. The method as claimed in claim 96, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to

said information is detected at said specified location.

106. The method as claimed in claim 96, further comprising:

(c) defining an otherwise operation, wherein said otherwise operation is performed when said matching operation is a failure.

107. The method as claimed in claim 106, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

108. The method as claimed in claim 96, wherein said occurrence is a variable to be detected, and

wherein said matching operation is defined by a match instruction that comprises a var modifier that identifies said variable to be detected.

109. The method as claimed in claim 108, wherein said matching instruction comprises a variable constraint, and

wherein said matching operation is satisfied when said variable has at least a certain value identified by said variable constraint.

110. The method as claimed in claim 108, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said matching operation to determine whether or not said variable exists in only said current datagram.

111. The method as claimed in claim 109, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in only said current datagram.

112. The method as claimed in claim 108, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said matching operation to determine whether or not said variable exists in at least one process flow identified by said flow modifier.

113. The method as claimed in claim 109, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in at least one process flow identified by said flow modifier.

114. The method as claimed in claim 108, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said matching operation to determine whether or not said variable exists at a particular location identified by said at modifier.

115. The method as claimed in claim 114, wherein said particular location comprises a particular layer of a communication model.

116. The method as claimed in claim 114, wherein said particular location identifies at least one of (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

117. The method as claimed in claim 109, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value at a particular location identified by said at modifier.

118. The method as claimed in claim 117, wherein said particular location comprises a particular layer of a communication model.

119. The method as claimed in claim 117, wherein said particular location identifies at least one of (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and(4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

120. The method as claimed in claim 108, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said matching operation to determine whether or not said variable exists at or after a particular location identified by said offset modifier.

121. The method as claimed in claim 109, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value at or after a particular location identified by said offset modifier.

122. The method as claimed in claim 108, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said matching operation to determine whether or not said variable exists in data travelling in a particular direction identified by said direction modifier.

123. The method as claimed in claim 109, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in data travelling in a particular direction identified by said direction modifier.

124. The method as claimed in claim 96, wherein said occurrence is a predefined event to be detected, and

wherein said matching operation is defined by a match instruction that comprises

an event modifier that identifies said predefined event to be detected.

125. The method as claimed in claim 124,

wherein said method further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable equals said at least said particular value.

126. The method as claimed in claim 124,

wherein said method further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable does not equal said at least said particular value.

127. The method as claimed in claim 124,

wherein said method further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable falls within said at least said particular range.

128. The method as claimed in claim 124,

wherein said method further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable does not fall within said at least said particular range.

129. The method as claimed in claim 96, wherein said matching operation is described in at least an application layer of said communication model.

130. The method as claimed in claim 96, wherein said matching operation is described in at least a presentation layer of said communication model.

131. The method as claimed in claim 96, wherein said matching operation is described in at least a session layer of said communication model.

132. The method as claimed in claim 96, wherein said matching operation is described in at least a transport layer of said communication model.

133. A method for controlling a network system, comprising:

(a) defining a concurrent operation, wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed and wherein said first group of operations correspond to information transmitted over said network system in at least one of application, presentation, session, transport and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said concurrent operation.

134. The method as claimed in claim 133, wherein said concurrent operation ceases performing all unsuccessful operations of said first group of operations when one operation of said first group of operations is a success.

135. The method as claimed in claim 134, further comprising:

(c) defining a third operation that is performed if all of said first group of operations are not a success.

136. A method for controlling a network system, comprising:

(a) defining a first throw operation, wherein said first throw operation identifies resynchronization operation corresponding to said first throw operation and wherein said first throw operation corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model, and

wherein said first throw operation causes a specific operation corresponding to said resynchronization operation to be performed;

(b) controlling at least a portion of said network system based on said first throw operation and said resynchronization operation.

137. The method as claimed in claim 136, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction.

138. The method as claimed in claim 137, wherein said operation (b) comprises:

(b1) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed.

139. The method as claimed in claim 138, wherein said resynchronization

operation identifies said first exception name.

140. The method as claimed in claim 138, further comprising:

(c) defining a second throw operation, wherein said second throw operation identifies a second exception name corresponding to said second throw instruction, and

wherein said resynchronization operation identifies said first exception name and said second exception name, and

wherein said operation (b1) comprises:

(b1a) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

(b1b) resynchronizing to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.

141. An apparatus for controlling a network system, comprising:

an interface coupled to a network of said network system; and

a processor that processes a first event that occurs in said network system, wherein said first event is defined via software and wherein said processor controls at least a portion of said network system when said first event is processed and wherein said first event corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model.

142. The apparatus as claimed in claim 141, wherein said first event is defined

in at least the network layer of said communication model.

143. The method as claimed in claim 141, wherein said first event is defined in at least the transport layer of said communication model.

144. The method as claimed in claim 141, wherein said first event is defined in at least the session layer of said communication model.

145. The method as claimed in claim 141, wherein said first event is defined in at least the presentation layer of said communication model.

146. The method as claimed in claim 141, wherein said first event is defined in at least the application layer of said communication model.

147. The apparatus as claimed in claim 141, wherein said processor processes a second event that is defined via said software, and

wherein said second event is defined at least partially with said first event.

148. The apparatus as claimed in claim 141, wherein said processor processes a second event that is defined via said software, and

wherein said first event is at least partially defined with said second event.

149. The apparatus as claimed in claim 141, wherein said processor performs a matching operation that is defined via said software, wherein said matching operation detects an occurrence of a variable in information transmitted over said network and

wherein said variable is identified in said matching operation, and wherein said first event is at least partially defined with said matching operation.

150. The apparatus as claimed in claim 149, wherein said matching operation causes said processor to identify whether or not a series of data within a process flow corresponds to said variable.

151. The apparatus as claimed in claim 149, wherein said matching operation causes said processor to identify whether or not an occurrence of a predefined event corresponds to said variable, and

wherein said matching operation causes said processor to determine that said predefined event has occurred when said predefined event corresponds to said variable.

152. The apparatus as claimed in claim 149, wherein said processor repeatedly performing said matching operation within a process flow.

153. The apparatus as claimed in claim 152, wherein said processor discontinues performance of said matching operation when said occurrence of said variable is detected.

154. The apparatus as claimed in claim 152, wherein said processor updates whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein processor deems said matching operation to be said success when said matching operation detects said occurrence of said variable, and

wherein said processor deems said matching operation to be said failure when said matching operation does not detect said occurrence of said variable.

155. The apparatus as claimed in claim 149, wherein said processor performs said matching operation on a plurality of process flows.

156. The apparatus as claimed in claim 149, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said processor deems said matching operation to be successful only if said occurrence of said variable in said information is detected after said specified location.

157. The apparatus as claimed in claim 149, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said processor deems said matching operation to be successful only if said occurrence of said variable in said information is detected at said specified location.

158. The apparatus as claimed in claim 149, wherein said processor performs an otherwise operation, wherein said otherwise operation is defined via software, wherein said otherwise operation is performed by said processor when said matching operation is a failure, and wherein said first event is at least partially defined with said otherwise operation.

159. The apparatus as claimed in claim 158, wherein said processor performs a throw operation that is defined via software and that identifies a resync operation corresponding to said throw operation,

wherein said throw operation causes said processor to perform a specific operation corresponding to said resync operation.

160. The apparatus as claimed in claim 158, wherein said processor performs said otherwise operation immediately after said matching operation when said matching operation is said failure.

161. The apparatus as claimed in claim 141, wherein said processor performs a concurrent operation that is defined via said software,

wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed by said processor, and

wherein said first event is at least partially defined with said concurrent operation.

162. The apparatus as claimed in claim 161, wherein said processor ceases performing all unsuccessful operations of said first group of operations within said concurrent operation when one operation of said first group of operations is a success.

163. The apparatus as claimed in claim 162, wherein said processor performs a third operation defined via said software when all of said first group of operations are not a success, and

wherein said first event is at least partially defined with said third operation.

164. The apparatus as claimed in claim 149, wherein the matching operation is defined with a modifier that identifies a specific location in said information where said

occurrence of said variable is to be detected by said processor.

165. The apparatus as claimed in claim 164, wherein said modifier identifies at least one of (1) a starting offset of an Internet protocol header in a current datagram transmitted on said network system, (2) a starting offset of a payload in a current datagram transmitted on said network system, (3) a starting offset of a transport header in a current datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a current process flow transmitted on said network system.

166. The apparatus as claimed in claim 141, wherein a timer is defined via said software, and

wherein said timer causes said processor to count time.

167. The apparatus as claimed in claim 166, wherein said timer is capable of instructing said processor to count said time in a selected time unit,

wherein said selected time unit is selected from one of a plurality of available predefined time units, and

wherein said timer is defined to instruct said processor to count said time in said selected time unit.

168. The apparatus as claimed in claim 167, wherein one of said predefined time units comprises at least one of milliseconds and seconds.

169. The apparatus as claimed in claim 166, wherein said processor decrements

a current count value of said timer until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said processor ceases counting and generates an indication that said current count value equals said predefined count value.

170. The apparatus as claimed in claim 166, wherein said processor increments a current count value of said timer until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said processor ceases counting and generates an indication that said current count value equals said predefined count value.

171. The apparatus as claimed in claim 166, wherein said timer is capable instructing said processor to count said time in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said timer is incremented and a second direction in which said count value is decremented, and

wherein said timer instructs said processor to count said time in said selected direction.

172. The apparatus as claimed in claim 171, wherein, when said count value equals a predefined count value, said processor ceases counting and generates an indication that said count value equals said predefined count value.

173. The apparatus as claimed in claim 166, wherein said processor performs a timer start operation that is defined via said software and that instructs said processor to start counting in accordance with said timer.

174. The apparatus as claimed in claim 166, wherein said processor performs a timer stop operation that is defined via said software and that instructs said processor to stop counting.

175. The apparatus as claimed in claim 166, wherein said processor performs a timer pause operation that is defined via said software and that instructs said processor to suspend counting.

176. The apparatus as claimed in claim 175, wherein said processor performs a timer resume operation that is defined via software and that instructs said processor to resume counting after said processor has suspended counting.

177. The apparatus as claimed in claim 141, wherein a meter is defined via said software, and

wherein said meter instructs said processor to count quantities of a selected quantity unit.

178. The apparatus as claimed in claim 177, wherein said selected quantity unit is selected from one of a plurality of available predefined quantity units.

179. The apparatus as claimed in claim 178, wherein one of said predefined

quantity units comprises at least one of bits of data and bytes of data.

180. The apparatus as claimed in claim 178, wherein one of said predefined quantity units comprises numbers of events.

181. The apparatus as claimed in claim 178, wherein one of said predefined quantity units comprises numbers of packets.

182. The apparatus as claimed in claim 178, wherein said one of said predefined quantity units comprises numbers of process flows.

183. The apparatus as claimed in claim 177, wherein said meter instructs said processor to count said quantities of said selected quantity unit in a plurality of process flows.

184. The apparatus as claimed in claim 177, wherein said meter instructs said processor to decrement a current count value of said meter until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said processor ceases counting and generates an indication that said current count value equals said predefined count value.

185. The apparatus as claimed in claim 177, wherein said meter instructs said processor to increment a current count value until said current count value of said meter equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said processor ceases counting and generates an indication that said current count value equals said predefined count value.

186. The apparatus as claimed in claim 177, wherein said meter is capable of instructing said processor to count said quantities in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said meter is incremented and a second direction in which said count value is decremented, and

wherein said meter instructs said processor to count said quantities in said selected direction.

187. The apparatus as claimed in claim 186, wherein, when said count value equals a predefined count value, said processor ceases counting and generates an indication that said count value equals said predefined count value.

188. The apparatus as claimed in claim 177, wherein said processor performs a meter start operation that is defined via said software and that instructs said processor to start counting in accordance with said meter.

189. The apparatus as claimed in claim 177, wherein said processor performs a meter stop operation that is defined via said software and that instructs said processor to stop counting.

190. The apparatus as claimed in claim 177, wherein said processor performs a

meter pause operation that is defined via said software and that instructs said processor to suspend counting.

191. The apparatus as claimed in claim 190, wherein said processor performs a meter resume operation that is defined via said software and that instructs said processor to resume counting after said processor has suspended counting.

192. The apparatus as claimed in claim 141, wherein said processor performs a mergeFlow operation which is defined via software, which adds a current process flow to at least one existing process flow, and which allocates joint resources of a computer system for said current process flow and said at least one existing process flow,

wherein said first event is at least partially defined with said mergeFlow operation.

193. The apparatus as claimed in claim 192, wherein said mergeFlow operation instructs said processor to assign a unique flow index number to said current process flow.

194. The apparatus as claimed in claim 193, wherein said flow index number of said current process flow is different than a flow index number of said at least one existing process flow.

195. The apparatus as claimed in claim 149, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed by said processor when said match

operation is a success, and

wherein said first event completes execution when said dedicated instruction is executed by said processor.

196. The apparatus as claimed in claim 195, wherein the dedicated instruction indicates a successful completion of said first event.

197. The apparatus as claimed in claim 149, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed by said processor when said match operation is unsuccessful, and

wherein said first event completes execution when said dedicated instruction is executed by said processor.

198. The apparatus as claimed in claim 197, wherein the dedicated instruction indicates an unsuccessful completion of said first event.

199. The apparatus as claimed in claim 196, wherein said processor generates successful completion data when said first event is successfully completed, and

wherein at least a second event defined via software instructs said processor to utilize said successful completion data to determine whether or not said first event has been successfully completed.

200. The apparatus as claimed in claim 198, wherein said processor generates unsuccessful completion data when said first event is not successfully completed, and

wherein at least a second event defined via software instructs said processor to utilize said unsuccessful completion data to determine whether or not said first event has been successfully completed.

201. The apparatus as claimed in claim 141, wherein a variable is defined as at least part of said first event,

wherein said first event instructs said processor to determine an alias for said variable, and

wherein said variable is capable of being referenced via said alias.

202. The apparatus as claimed in claim 201, wherein said processor processes a second event, wherein a variable reference to said alias is part of said second event and wherein said second event instructs said processor to utilize said variable via said variable reference.

203. The apparatus as claimed in claim 201, wherein said variable has a plurality of subparts, and

wherein said alias is defined as a subset of said variable containing less than all of said subparts such that said variable reference corresponds to only said subset.

204. The apparatus as claimed in claim 203, wherein said subparts are subfields of said variable.

205. The apparatus as claimed in claim 141, wherein a variable is defined as at least part of said first event,

wherein said processor constrains said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable equals said at least said particular value.

206. The apparatus as claimed in claim 205, wherein said particular operation is a match operation.

207. The apparatus as claimed in claim 205, wherein said particular operation is an event definition.

208. The apparatus as claimed in claim 141, wherein a variable is defined as at least part of said first event,

wherein said processor constrains said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable does not equal said at least said particular value.

209. The apparatus as claimed in claim 208, wherein said particular operation is a match operation.

210. The apparatus as claimed in claim 208, wherein said particular operation is an event definition.

211. The apparatus as claimed in claim 141, wherein a variable is defined as at least part of said first event,

wherein said processor constrains said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable is within said particular range.

212. The apparatus as claimed in claim 211, wherein said particular operation is a match operation.

213. The apparatus as claimed in claim 211, wherein said particular operation is an event definition.

214. The apparatus as claimed in claim 141, wherein a variable is defined as at least part of said first event,

wherein said processor constrains said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation, and

wherein said particular operation is a success if said variable is not within said particular range.

215. The apparatus as claimed in claim 214, wherein said particular operation is a match operation.

216. The apparatus as claimed in claim 214, wherein said particular operation is an event definition.

217. The apparatus as claimed in claim 202, wherein said alias is defined as a particular subfield of said variable,

wherein said processor constrains said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable reference equals said particular value.

218. The apparatus as claimed in claim 217, wherein said particular operation is a match operation.

219. The apparatus as claimed in claim 217, wherein said particular operation is an event definition.

220. The apparatus as claimed in claim 202, wherein said alias is defined as a particular subfield of said variable,

wherein said processor constrains said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable reference does not equal said particular value.

221. The apparatus as claimed in claim 220, wherein said particular operation is a match operation.

222. The apparatus as claimed in claim 220, wherein said particular operation is an event definition.

223. The apparatus as claimed in claim 202, wherein said alias is defined as a particular subfield of said variable,

wherein said processor constrains said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable reference is within said particular range.

224. The apparatus as claimed in claim 223, wherein said particular operation is a match operation.

225. The apparatus as claimed in claim 223, wherein said particular operation is an event definition.

226. The apparatus as claimed in claim 202, wherein said alias is defined as a

particular subfield of said variable,

wherein said processor constrains said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation, and

wherein said processor determines that said particular operation is a success if said variable reference is not within said particular range.

227. The apparatus as claimed in claim 226, wherein said particular operation is a match operation.

228. The apparatus as claimed in claim 226, wherein said particular operation is an event definition.

229. The apparatus as claimed in claim 141, wherein said first event is described in at least a first layer of said communication model.

230. The apparatus as claimed in claim 229, wherein said first layer is one of said application layer, said presentation layer, said session layer, said transport layer, and said network layer of said communication model.

231. The apparatus as claimed in claim 230, wherein said first event is transformed into actions in said first layer of said communication model.

232. The apparatus as claimed in claim 230, wherein said first event is

transformed into actions in a second layer of said communication model, wherein said second layer is lower in said communication model than said first layer.

233. The apparatus as claimed in claim 141, wherein said processor performs a first throw operation, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction,

wherein said first event at least partially with said first throw instruction, and

wherein said processor performs a resynchronization operation.

234. The apparatus as claimed in claim 233, wherein said processor resynchronizes to an instruction immediately following said resynchronization operation when said first throw operation is performed.

235. The apparatus as claimed in claim 234, wherein said resynchronization operation identifies said first exception name.

236. The apparatus as claimed in claim 234, wherein said processor performs a second throw operation,

wherein said second throw operation identifies a second exception name corresponding to said second throw instruction,

wherein said resynchronization operation identifies said first exception name and said second exception name,

wherein said processor resynchronizes to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

wherein said processor resynchronizes to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.

237. An apparatus for controlling a network system, comprising:

an interface coupled to a network of said network system; and

a processor that processes a matching operation that occurs in said network system,

wherein said matching operation is defined via software and instructs said processor to detect an occurrence corresponding to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model,

wherein said occurrence is identified in said matching operation, and

wherein said processor controls at least a portion of said network system based on said matching operation.

238. The apparatus as claimed in claim 237, wherein a first event is defined via said software and is defined at least partially with said matching operation.

239. The apparatus as claimed in claim 237, wherein said matching operation instructs said processor to identify whether or not a series of data within a process flow corresponds to said occurrence.

240. The apparatus as claimed in claim 237, wherein said occurrence corresponds to a predefined event, and

wherein said matching operation instructs said processor to identify whether or not said predefined event has occurred.

241. The apparatus as claimed in claim 237, wherein said processor repeatedly performs said matching operation within a process flow.

242. The apparatus as claimed in claim 241, wherein said processor discontinues performance of said matching operation when said occurrence is detected.

243. The apparatus as claimed in claim 241, wherein said processor updates whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said occurrence is detected, and

wherein said matching operation is said failure when said occurrence is not detected.

244. The apparatus as claimed in claim 237, wherein said matching operation is performed on a plurality of process flows.

245. The apparatus as claimed in claim 237, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected after said specified location.

246. The apparatus as claimed in claim 237, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected at said specified location.

247. The apparatus as claimed in claim 237, wherein said processor performs an otherwise operation, and

wherein said otherwise operation is performed when said matching operation is a failure.

248. The apparatus as claimed in claim 247, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

249. The apparatus as claimed in claim 237, wherein said occurrence is a variable to be detected, and

wherein said matching operation is defined by a match instruction that comprises a var modifier that identifies said variable to be detected.

250. The apparatus as claimed in claim 249, wherein said matching instruction comprises a variable constraint, and

wherein said matching operation is satisfied when said variable has at least a certain value identified by said variable constraint.

251. The apparatus as claimed in claim 249, wherein said match instruction

comprises an immediate modifier, and

wherein said immediate modifier causes said processor to determine whether or not said variable exists in only said current datagram.

252. The apparatus as claimed in claim 250, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said processor to determine whether or not said variable exists and has at least said certain value in only said current datagram.

253. The apparatus as claimed in claim 249, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said processor to determine whether or not said variable exists in at least one process flow identified by said flow modifier.

254. The apparatus as claimed in claim 250, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said processor to determine whether or not said variable exists and has at least said certain value in at least one process flow identified by said flow modifier.

255. The apparatus as claimed in claim 249, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said processor to determine whether or not said variable exists at a particular location identified by said at modifier.

256. The apparatus as claimed in claim 255, wherein said particular location comprises a particular layer of a communication model.

257. The apparatus as claimed in claim 255, wherein said particular location identifies at least one of (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

258. The apparatus as claimed in claim 250, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said processor to determine whether or not said variable exists and has at least said certain value at a particular location identified by said at modifier.

259. The apparatus as claimed in claim 258, wherein said particular location comprises a particular layer of a communication model.

260. The apparatus as claimed in claim 258, wherein said particular location identifies (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

261. The apparatus as claimed in claim 249, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said processor to determine whether or not said variable exists at or after a particular location identified by said offset modifier.

262. The apparatus as claimed in claim 250, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said processor to determine whether or not said variable exists and has at least said certain value at or after a particular location identified by said offset modifier.

263. The apparatus as claimed in claim 249, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said processor to determine whether or not said variable exists in data travelling in a particular direction identified by said direction modifier.

264. The apparatus as claimed in claim 250, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said processor to determine whether or not said variable exists and has at least said certain value in data travelling in a particular direction identified by said direction modifier.

265. The apparatus as claimed in claim 237, wherein said occurrence is a

predefined event to be detected, and

wherein said matching operation is defined by a match instruction that comprises an event modifier that identifies said predefined event to be detected.

266. The apparatus as claimed in claim 265, wherein a variable is defined as at least part of said predefined event,

wherein a variable constraint is defined in said matching instruction,

wherein said variable constraint instructs said processor constrains said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said processor determines that said matching operation is a success if said variable equals said at least said particular value.

267. The apparatus as claimed in claim 265, wherein a variable is defined as at least part of said predefined event,

wherein a variable constraint is defined in said matching instruction,

wherein said variable constraint instructs said processor to constrain said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said processor determining that said matching operation is a success if said variable does not equal said at least said particular value.

268. The apparatus as claimed in claim 265, wherein a variable as at least part of said predefined event,

wherein a variable constraint is defined in said matching instruction,

wherein said variable constraint instructs said processor to constrain said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said processor determines that said matching operation is a success if said variable falls within said at least said particular range.

269. The apparatus as claimed in claim 265, wherein a variable is defined as at least part of said predefined event,

wherein a variable constraint is defined in said matching instruction,

wherein said variable constraint instructs said processor to constrain said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said processor determines that said matching operation is a success if said variable does not fall within said at least said particular range.

270. The apparatus as claimed in claim 237, wherein said matching operation is described in at least said application layer of said communication model.

271. The apparatus as claimed in claim 237, wherein said matching operation is described in at least said presentation layer of said communication model.

272. The apparatus as claimed in claim 237, wherein said matching operation is described in at least said session layer of said communication model.

273. The apparatus as claimed in claim 237, wherein said matching operation is described in at least said transport layer of said communication model.

274. The apparatus as claimed in claim 237, wherein said matching operation is described in at least said network layer of said communication model.

275. An apparatus for controlling a network system, comprising:

an interface coupled to a network of said network system; and

a processor that processes a concurrent operation,

wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently processed by said processor and wherein said first group of operations corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model, and

wherein said processor controls at least a portion of said network system based on said concurrent operation.

276. The apparatus as claimed in claim 275, wherein said processor ceases performing all unsuccessful operations of said first group of operations when one operation of said first group of operations is a success.

277. The apparatus as claimed in claim 276, wherein said processor processes a third operation that is performed if all of said first group of operations are not a success.

278. An apparatus for controlling a network system, comprising:

an interface coupled to a network of said network system; and

a processor that performs a first throw operation,

wherein said first throw operation identifies resynchronization operation corresponding to said first throw operation and wherein said first throw operation corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of said communication model,

wherein said first throw operation causes said processor to perform a specific operation corresponding to said resynchronization operation, and

wherein said processor controls at least a portion of said network system based on said first throw operation and said resynchronization operation.

279. The apparatus as claimed in claim 278, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction.

280. The apparatus as claimed in claim 279, wherein said processor resynchronizes to an instruction immediately following said resynchronization operation when said first throw operation is performed.

281. The apparatus as claimed in claim 280, wherein said resynchronization operation identifies said first exception name.

282. The apparatus as claimed in claim 280, wherein said processor performs a second throw operation,

wherein said second throw operation identifies a second exception name corresponding to said second throw instruction,

wherein said resynchronization operation identifies said first exception name and said second exception name,

wherein said processor resynchronizes to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

wherein said processor resynchronizes to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.

283. A software program contained in a computer readable medium containing instructions for causing a processor to perform a routine, comprising:

(a) defining a first event that occurs in said network system, wherein said event is defined via software and wherein said first event corresponds to information transmitted over said network system in at least one of application, presentation, session, transport and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said first event.

284. The software program as claimed in claim 283, wherein said first event is defined in at least the network layer of said communication model.

285. The software program as claimed in claim 283, wherein said first event is defined in at least the transport layer of said communication model.

286. The software program as claimed in claim 283, wherein said first event is defined in at least the session layer of said communication model.

287. The software program as claimed in claim 283, wherein said first event is defined in at least the presentation layer of said communication model.

288. The software program as claimed in claim 283, wherein said first event is defined in at least the application layer of said communication model.

289. The software program as claimed in claim 283, the routine further comprising:

(c) defining a second event via said software, wherein said second event is defined at least partially with said first event.

290. The software program as claimed in claim 283, wherein said operation (a) comprises:

(a1) defining a second event via said software; and

(a2) defining said first event at least partially with said second event.

291. The software program as claimed in claim 283, wherein said operation (a) comprises:

(a1) defining a matching operation via said software, wherein said matching operation detects an occurrence of a variable in information transmitted over said network system, wherein said variable is identified in said matching operation; and

(a2) defining said first event at least partially with said matching operation.

292. The software program as claimed in claim 291, wherein said matching operation identifies whether or not a series of data within a process flow corresponds to said variable.

293. The software program as claimed in claim 291, wherein said matching operation identifies whether or not an occurrence of a predefined event corresponds to said variable, and

wherein said matching operation determines that said predefined event has occurred when said predefined event corresponds to said variable.

294. The software program as claimed in claim 291, the routine further comprising:

(c) repeatedly performing said matching operation within a process flow.

295. The software program as claimed in claim 294, the routine further comprising:

(d) discontinuing performance of said matching operation when said matching operation detects said occurrence of said variable.

296. The software program as claimed in claim 294, wherein said operation (c) comprises:

(c1) updating whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said matching operation detects said occurrence of said variable, and

wherein said matching operation is said failure when said matching operation does not detect said occurrence of said variable.

297. The software program as claimed in claim 291, wherein said matching operation is performed on a plurality of process flows.

298. The software program as claimed in claim 291, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence of said variable in said information is detected after said specified location.

299. The software program as claimed in claim 291, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence of said variable in said information is detected at said specified location.

300. The software program as claimed in claim 291, wherein said operation (a) further comprises:

(a3) defining an otherwise operation, wherein said otherwise operation is performed when said matching operation is a failure; and

(a4) defining said first event at least partially with said otherwise operation.

301. The software program as claimed in claim 300, wherein said operation (a3)

comprises:

(a3a) defining a throw operation,

wherein said throw operation identifies resync operation corresponding to said throw operation, and

wherein said throw operation causes a specific operation corresponding to said resync operation to be performed.

302. The software program as claimed in claim 300, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

303. The software program as claimed in claim 283, wherein said operation (a) comprises:

(a1) defining a concurrent operation, wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed; and

(a2) defining said first event at least partially with said concurrent operation.

304. The software program as claimed in claim 303, wherein said concurrent operation ceases performing all unsuccessful operations of said first group of operations when one operation of said first group of operations is a success.

305. The software program as claimed in claim 304, wherein said operation (a) comprises:

(a3) defining a third operation that is performed if all of said first group of

operations are not a success; and

(a4) defining said first event at least partially with said third operation.

306. The software program as claimed in claim 291, wherein the matching operation is defined with a modifier that identifies a specific location in said information where said occurrence of said variable is to be detected.

307. The software program as claimed in claim 306, wherein said modifier identifies at least one of (1) a starting offset of an Internet protocol header in a current datagram transmitted on said network system, (2) a starting offset of a payload in a current datagram transmitted on said network system, (3) a starting offset of a transport header in a current datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a current process flow transmitted on said network system.

308. The software program as claimed in claim 283, the routine further comprising:

(c) defining a timer via said software, wherein said timer counts time.

309. The software program as claimed in claim 308, wherein said timer is capable counting time in a selected time unit,

wherein said selected time unit is selected from one of a plurality of available predefined time units, and

wherein said timer is defined to count time in said selected time unit.

310. The software program as claimed in claim 309, wherein one of said predefined time units comprises one of milliseconds and seconds.

311. The software program as claimed in claim 308, wherein said timer decrements a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said timer ceases counting and generates an indication that said current count value equals said predefined count value.

312. The software program as claimed in claim 308, wherein said timer increments a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said timer ceases counting and generates an indication that said current count value equals said predefined count value.

313. The software program as claimed in claim 308, wherein said timer is capable counting time in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said timer is incremented and a second direction in which said count value is decremented, and

wherein said timer is defined to count time in said selected direction.

314. The software program as claimed in claim 313, wherein, when said count value equals a predefined count value, said timer ceases counting and generates an

indication that said count value equals said predefined count value.

315. The software program as claimed in claim 308, the routine further comprising:

(d) providing a timer start operation that instructs said timer to start counting.

316. The software program as claimed in claim 308, the routine further comprising:

(d) providing a timer stop operation that instructs said timer to stop counting.

317. The software program as claimed in claim 308, the routine further comprising:

(d) providing a timer pause operation that instructs said timer to suspend counting.

318. The software program as claimed in claim 308, the routine further comprising:

(e) providing a timer resume operation that instructs said timer to resume counting after said timer has suspended counting.

319. The software program as claimed in claim 283, the routine further comprising:

(c) defining a meter via said software, wherein said meter counts quantities of a selected quantity unit.

320. The software program as claimed in claim 319, wherein said selected quantity unit is selected from one of a plurality of available predefined quantity units.

321. The software program as claimed in claim 320, wherein one of said predefined quantity units comprises one of bits of data and bytes of data.

322. The software program as claimed in claim 320, wherein one of said predefined quantity units comprises numbers of events.

323. The software program as claimed in claim 320, wherein one of said predefined quantity units comprises numbers of packets.

324. The software program as claimed in claim 320, wherein said one of said predefined quantity units comprises numbers of process flows.

325. The software program as claimed in claim 319, wherein said meter counts said quantities of said selected quantity unit in a plurality of process flows.

326. The software program as claimed in claim 319, wherein said meter decrements a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said meter ceases counting and generates an indication that said current count value equals said predefined count value.

327. The software program as claimed in claim 319, wherein said meter increments a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said meter ceases counting and generates an indication that said current count value equals said predefined count value.

328. The software program as claimed in claim 319, wherein said meter is capable counting said quantities in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said meter is incremented and a second direction in which said count value is decremented, and

wherein said meter is defined to count said quantities in said selected direction.

329. The software program as claimed in claim 328, wherein, when said count value equals a predefined count value, said meter ceases counting and generates an indication that said count value equals said predefined count value.

330. The software program as claimed in claim 319, the routine further comprising:

(d) providing a meter start operation that instructs said meter to start counting.

331. The software program as claimed in claim 319, the routine further comprising:

(d) providing a meter stop operation that instructs said meter to stop counting.

332. The software program as claimed in claim 319, the routine further comprising:

(d) providing a meter pause operation that instructs said meter to suspend counting.

333. The software program as claimed in claim 332, the routine further comprising:

(e) providing a meter resume operation that instructs said meter to resume counting after said meter has suspended counting.

334. The software program as claimed in claim 283, wherein said operation (a) comprises:

(a1) defining an mergeFlow operation which adds a current process flow to at least one existing process flow and which allocates joint resources of a computer system for said current process flow and said at least one existing process flow; and

(a2) defining said first event at least partially with said mergeFlow operation.

335. The software program as claimed in claim 334, wherein said mergeFlow operation assigns a unique flow index number to said current process flow.

336. The software program as claimed in claim 335, wherein said flow index number of said current process flow is different than a flow index number of said at least one existing process flow.

337. The software program as claimed in claim 291, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed when said match operation is a success, and

wherein said first event completes execution when said dedicated instruction is executed.

338. The software program as claimed in claim 337, wherein the dedicated instruction indicates a successful completion of said first event.

339. The software program as claimed in claim 291, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed when said match operation is unsuccessful, and

wherein said first event completes execution when said dedicated instruction is executed.

340. The software program as claimed in claim 339, wherein the dedicated instruction indicates an unsuccessful completion of said first event.

341. The software program as claimed in claim 338, wherein said first event generates successful completion data when said first event is successfully completed, and

wherein at least a second event utilizes said successful completion data to determine whether or not said first event has been successfully completed.

342. The software program as claimed in claim 340, wherein said first event generates unsuccessful completion data when said first event is not successfully completed, and

wherein at least a second event utilizes said unsuccessful completion data to determine whether or not said first event has been successfully completed.

343. The software program as claimed in claim 283, wherein said operation (a) comprises:

(a1) defining a variable as part of said first event; and

(a2) defining an alias for said variable as part of said first event, wherein said variable is capable of being referenced via said alias.

344. The software program as claimed in claim 343, further comprising:

(c) defining a second event, wherein a variable reference to said alias is part of said second event and wherein said second event can utilize said variable via said variable reference.

345. The software program as claimed in claim 343, wherein said variable has a plurality of subparts, and

wherein said alias is defined as a subset of said variable containing less than all of said subparts such that said variable reference corresponds to only said subset.

346. The software program as claimed in claim 345, wherein said subparts are subfields of said variable.

347. The software program as claimed in claim 283,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable equals said at least said particular value.

348. The software program as claimed in claim 347, wherein said particular operation is a match operation.

349. The software program as claimed in claim 347, wherein said particular operation is an event definition.

350. The software program as claimed in claim 283,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable does not equal said at least said particular value.

351. The software program as claimed in claim 350, wherein said particular operation is a match operation.

352. The software program as claimed in claim 350, wherein said particular operation is an event definition.

353. The software program as claimed in claim 283,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable is within said particular range.

354. The software program as claimed in claim 353, wherein said particular operation is a match operation.

355. The software program as claimed in claim 353, wherein said particular operation is an event definition.

356. The software program as claimed in claim 283,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable is not within said particular range.

357. The software program as claimed in claim 356, wherein said particular operation is a match operation.

358. The software program as claimed in claim 356, wherein said particular operation is an event definition.

359. The software program as claimed in claim 344, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable reference equals said particular value.

360. The software program as claimed in claim 359, wherein said particular operation is a match operation.

361. The software program as claimed in claim 359, wherein said particular operation is an event definition.

362. The software program as claimed in claim 344, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable reference does not equal said particular value.

363. The software program as claimed in claim 362, wherein said particular operation is a match operation.

364. The software program as claimed in claim 362, wherein said particular operation is an event definition.

365. The software program as claimed in claim 344, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation;

and

(e) determining that said particular operation is a success if said variable reference is within said particular range.

366. The software program as claimed in claim 365, wherein said particular operation is a match operation.

367. The software program as claimed in claim 365, wherein said particular operation is an event definition.

368. The software program as claimed in claim 344, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable reference is not within said particular range.

369. The software program as claimed in claim 368, wherein said particular operation is a match operation.

370. The software program as claimed in claim 368, wherein said particular operation is an event definition.

371. The software program as claimed in claim 283, wherein said first event is described in at least a first layer of said communication model.

372. The software program as claimed in claim 371, wherein said first layer is one of said application layer, said presentation layer, said session layer, said transport layer, and said network layer of said communication model.

373. The software program as claimed in claim 372, the routine further comprising:

(c) transforming said first event into actions in said first layer of said communication model.

374. The software program as claimed in claim 372, the routine further comprising:

(c) transforming said first event into actions in a second layer of said communication model, wherein said second layer is lower in said communication model than said first layer.

375. The software program as claimed in claim 283, wherein said operation (a) comprises:

(a1) defining a first throw operation, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction; and

(a2) defining said first event at least partially with said first throw instruction, and

wherein said routine further comprises:

(d) defining a resynchronization operation.

376. The software program as claimed in claim 375, the routine further comprising:

(e) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed.

377. The software program as claimed in claim 376, wherein said resynchronization operation identifies said first exception name.

378. The software program as claimed in claim 376, the routine further comprising:

(f) defining a second throw operation, wherein said second throw operation identifies a second exception name corresponding to said second throw instruction, and

wherein said resynchronization operation identifies said first exception name and said second exception name, and

wherein said operation (e) comprises:

(e1) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

(e2) resynchronizing to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.

379. A software program contained in a computer readable medium containing instructions for causing a processor to perform a routine, comprising:

(a) defining a matching operation that occurs in said network system, wherein said matching operation is defined via software and detects an occurrence corresponding to information transmitted over said network system and corresponding to information transmitted over said network system in at least one of application, presentation, session, transport and network layers of a communication model, and

wherein said occurrence is identified in said matching operation; and

(b) controlling at least a portion of said network system based on said matching operation.

380. The software program as claimed in claim 379, wherein said operation (a) comprises:

(a1) defining a first event via said software; and

(a2) defining said first event at least partially with said matching operation.

381. The software program as claimed in claim 380, wherein said matching operation identifies whether or not a series of data within a process flow corresponds to said occurrence.

382. The software program as claimed in claim 380, wherein said occurrence corresponds to a predefined event, and

wherein said matching operation identifies whether or not said predefined event has occurred.

383. The software program as claimed in claim 380, the routine further comprising:

(c) repeatedly performing said matching operation within a process flow.

384. The software program as claimed in claim 383, the routine further comprising:

(d) discontinuing performance of said matching operation when said matching operation detects said occurrence.

385. The software program as claimed in claim 383, wherein said operation (c) comprises:

(c1) updating whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said matching operation detects said occurrence, and

wherein said matching operation is said failure when said matching operation does not detect said occurrence.

386. The software program as claimed in claim 379, wherein said matching operation is performed on a plurality of process flows.

387. The software program as claimed in claim 379, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected after said specified location.

388. The software program as claimed in claim 379, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected at said specified location.

389. The software program as claimed in claim 379, the routine further comprising:

(c) defining an otherwise operation, wherein said otherwise operation is performed when said matching operation is a failure.

390. The software program as claimed in claim 379, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

391. The software program as claimed in claim 379, wherein said occurrence is a variable to be detected, and

wherein said matching operation is defined by a match instruction that comprises a var modifier that identifies said variable to be detected.

392. The software program as claimed in claim 391, wherein said matching instruction comprises a variable constraint, and

wherein said matching operation is satisfied when said variable has at least a certain value identified by said variable constraint.

393. The software program as claimed in claim 391, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said matching operation to determine whether or not said variable exists in only said current datagram.

394. The software program as claimed in claim 392, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in only said current datagram.

395. The software program as claimed in claim 391, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said matching operation to determine whether or not said variable exists in at least one process flow identified by said flow modifier.

396. The software program as claimed in claim 392, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in at least one process flow identified by said flow modifier.

397. The software program as claimed in claim 391, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said matching operation to determine whether or

not said variable exists at a particular location identified by said at modifier.

398. The software program as claimed in claim 397, wherein said particular location comprises a particular layer of a communication model.

399. The software program as claimed in claim 397, wherein said particular location identifies at least one of (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and(4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

400. The software program as claimed in claim 392, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value at a particular location identified by said at modifier.

401. The software program as claimed in claim 400, wherein said particular location comprises a particular layer of a communication model.

402. The software program as claimed in claim 400, wherein said particular location identifies at least one of (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in

a datagram transmitted on said network system, and(4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

403. The software program as claimed in claim 291, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said matching operation to determine whether or not said variable exists at or after a particular location identified by said offset modifier.

404. The software program as claimed in claim 392, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value at or after a particular location identified by said offset modifier.

405. The software program as claimed in claim 391, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said matching operation to determine whether or not said variable exists in data travelling in a particular direction identified by said direction modifier.

406. The software program as claimed in claim 392, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in data travelling in a

particular direction identified by said direction modifier.

407. The software program as claimed in claim 379, wherein said occurrence is a predefined event to be detected, and

wherein said matching operation is defined by a match instruction that comprises an event modifier that identifies said predefined event to be detected.

408. The software program as claimed in claim 407,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable equals said at least said particular value.

409. The software program as claimed in claim 407,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular

value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable does not equal said at least said particular value.

410. The software program as claimed in claim 407,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable falls within said at least said particular range.

411. The software program as claimed in claim 407,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said

variable equal to said at least said particular range, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable does not fall within said at least said particular range.

412. The software program as claimed in claim 379, wherein said matching operation is described in at least an application layer of said communication model.

413. The software program as claimed in claim 379, wherein said matching operation is described in at least a presentation layer of said communication model.

414. The software program as claimed in claim 379, wherein said matching operation is described in at least a session layer of said communication model.

415. The software program as claimed in claim 379, wherein said matching operation is described in at least a transport layer of said communication model.

416. The software program as claimed in claim 379, wherein said matching operation is described in at least a network layer of said communication model.

417. A software program contained in a computer readable medium containing instructions for causing a processor to perform a routine, comprising:

(a) defining a concurrent operation, wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed and wherein said first group of operations

corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said concurrent operation.

418. The software program as claimed in claim 417, wherein said concurrent operation ceases performing all unsuccessful operations of said first group of operations when one operation of said first group of operations is a success.

419. The software program as claimed in claim 418, the routine further comprising:

(c) defining a third operation that is performed if all of said first group of operations are not a success.

420. A software program contained in a computer readable medium containing instructions for causing a processor to perform a routine, comprising:

(a) defining a first throw operation, wherein said first throw operation identifies resynchronization operation corresponding to said first throw operation, and

wherein said first throw operation causes a specific operation corresponding to said resynchronization operation to be performed and corresponding to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said first throw operation and said resynchronization operation.

421. The software program as claimed in claim 420, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction.

422. The software program as claimed in claim 421, wherein said operation (b) comprises:

(b1) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed.

423. The software program as claimed in claim 422, wherein said resynchronization operation identifies said first exception name.

424. The software program as claimed in claim 422, the routine further comprising:

(c) defining a second throw operation, wherein said second throw operation identifies a second exception name corresponding to said second throw instruction, and

wherein said resynchronization operation identifies said first exception name and said second exception name, and

wherein said operation (b1) comprises:

(b1a) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

(b1b) resynchronizing to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.

425. A development system for creating an application program, comprising:

a computer readable medium containing a software program having instructions; and

a compiler that compiles said instructions into network triggers for a network processor,

wherein said software program contains instructions to perform a routine, comprising:

(a) defining a first event that occurs in said network system, wherein said event is defined via software and corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said first event.

426. The development system as claimed in claim 425, the routine further comprising:

(c) defining a second event via said software, wherein said second event is defined at least partially with said first event.

427. The development system as claimed in claim 425, wherein said operation (a) comprises:

(a1) defining a second event via said software; and

(a2) defining said first event at least partially with said second event.

428. The development system as claimed in claim 425, wherein said operation (a) comprises:

(a1) defining a matching operation via said software, wherein said matching operation detects an occurrence of a variable in information transmitted over said network system, wherein said variable is identified in said matching operation; and

(a2) defining said first event at least partially with said matching operation.

429. The development system as claimed in claim 428, wherein said matching operation identifies whether or not a series of data within a process flow corresponds to said variable.

430. The development system as claimed in claim 428, wherein said matching operation identifies whether or not an occurrence of a predefined event corresponds to said variable, and

wherein said matching operation determines that said predefined event has occurred when said predefined event corresponds to said variable.

431. The development system as claimed in claim 428, the routine further comprising:

(c) repeatedly performing said matching operation within a process flow.

432. The development system as claimed in claim 431, the routine further comprising:

(d) discontinuing performance of said matching operation when said matching operation detects said occurrence of said variable.

433. The development system as claimed in claim 431, wherein said operation (c) comprises:

(c1) updating whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said matching operation detects said occurrence of said variable, and

wherein said matching operation is said failure when said matching operation does not detect said occurrence of said variable.

434. The development system as claimed in claim 428, wherein said matching operation is performed on a plurality of process flows.

435. The development system as claimed in claim 428, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence of said variable in said information is detected after said specified location.

436. The development system as claimed in claim 428, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence of said variable in said information is detected at said specified location.

437. The development system as claimed in claim 428, wherein said operation (a) further comprises:

(a3) defining an otherwise operation, wherein said otherwise operation is performed when said matching operation is a failure; and

(a4) defining said first event at least partially with said otherwise operation.

438. The development system as claimed in claim 437, wherein said operation (a3) comprises:

(a3a) defining a throw operation,

wherein said throw operation identifies resync operation corresponding to said throw operation, and

wherein said throw operation causes a specific operation corresponding to said resync operation to be performed.

439. The development system as claimed in claim 437, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

440. The development system as claimed in claim 425, wherein said operation (a) comprises:

(a1) defining a concurrent operation, wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed; and

(a2) defining said first event at least partially with said concurrent operation.

441. The development system as claimed in claim 440, wherein said concurrent operation ceases performing all unsuccessful operations of said first group of operations

when one operation of said first group of operations is a success.

442. The development system as claimed in claim 441, wherein said operation (a) comprises:

(a3) defining a third operation that is performed if all of said first group of operations are not a success; and

(a4) defining said first event at least partially with said third operation.

443. The development system as claimed in claim 428, wherein the matching operation is defined with a modifier that identifies a specific location in said information where said occurrence of said variable is to be detected.

444. The development system as claimed in claim 443, wherein said modifier identifies at least one of (1) a starting offset of an Internet protocol header in a current datagram transmitted on said network system, (2) a starting offset of a payload in a current datagram transmitted on said network system, (3) a starting offset of a transport header in a current datagram transmitted on said network system, and(4) a starting offset of a first payload in a series of concatenated payloads in a current process flow transmitted on said network system.

445. The development system as claimed in claim 425, the routine further comprising:

(c) defining a timer via said software, wherein said timer counts time.

446. The development system as claimed in claim 445, wherein said timer is

capable counting time in a selected time unit,

wherein said selected time unit is selected from one of a plurality of available predefined time units, and

wherein said timer is defined to count time in said selected time unit.

447. The development system as claimed in claim 446, wherein one of said predefined time units comprises at least one of milliseconds and seconds.

448. The development system as claimed in claim 445, wherein said timer decrements a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said timer ceases counting and generates an indication that said current count value equals said predefined count value.

449. The development system as claimed in claim 445, wherein said timer increments a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said timer ceases counting and generates an indication that said current count value equals said predefined count value.

450. The development system as claimed in claim 445, wherein said timer is capable counting time in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said timer is incremented and a second direction in which said count value is

decremented, and

wherein said timer is defined to count time in said selected direction.

451. The development system as claimed in claim 450, wherein, when said count value equals a predefined count value, said timer ceases counting and generates an indication that said count value equals said predefined count value.

452. The development system as claimed in claim 445, the routine further comprising:

(d) providing a timer start operation that instructs said timer to start counting.

453. The development system as claimed in claim 445, the routine further comprising:

(d) providing a timer stop operation that instructs said timer to stop counting.

454. The development system as claimed in claim 445, the routine further comprising:

(d) providing a timer pause operation that instructs said timer to suspend counting.

455. The development system as claimed in claim 454, the routine further comprising:

(e) providing a timer resume operation that instructs said timer to resume counting after said timer has suspended counting.

456. The development system as claimed in claim 425, the routine further comprising:

(c) defining a meter via said software, wherein said meter counts quantities of a selected quantity unit.

457. The development system as claimed in claim 456, wherein said selected quantity unit is selected from one of a plurality of available predefined quantity units.

458. The development system as claimed in claim 457, wherein one of said predefined quantity units comprises at least one of bits of data and bytes of data.

459. The development system as claimed in claim 457, wherein one of said predefined quantity units comprises numbers of events.

460. The development system as claimed in claim 457, wherein one of said predefined quantity units comprises numbers of packets.

461. The development system as claimed in claim 457, wherein said one of said predefined quantity units comprises numbers of process flows.

462. The development system as claimed in claim 456, wherein said meter counts said quantities of said selected quantity unit in a plurality of process flows.

463. The development system as claimed in claim 456, wherein said meter decrements a current count value until said current count value equals a predefined count

value, and

wherein, when said current count value equals said predefined count value, said meter ceases counting and generates an indication that said current count value equals said predefined count value.

464. The development system as claimed in claim 456, wherein said meter increments a current count value until said current count value equals a predefined count value, and

wherein, when said current count value equals said predefined count value, said meter ceases counting and generates an indication that said current count value equals said predefined count value.

465. The development system as claimed in claim 456, wherein said meter is capable counting said quantities in a selected direction,

wherein said selected direction is selected from a first direction in which a count value of said meter is incremented and a second direction in which said count value is decremented, and

wherein said meter is defined to count said quantities in said selected direction.

466. The development system as claimed in claim 465, wherein, when said count value equals a predefined count value, said meter ceases counting and generates an indication that said count value equals said predefined count value.

467. The development system as claimed in claim 456, the routine further comprising:

(d) providing a meter start operation that instructs said meter to start counting.

468. The development system as claimed in claim 456, the routine further comprising:

(d) providing a meter stop operation that instructs said meter to stop counting.

469. The development system as claimed in claim 456, the routine further comprising:

(d) providing a meter pause operation that instructs said meter to suspend counting.

470. The development system as claimed in claim 469, the routine further comprising:

(e) providing a meter resume operation that instructs said meter to resume counting after said meter has suspended counting.

471. The development system as claimed in claim 425, wherein said operation (a) comprises:

(a1) defining an mergeFlow operation which adds a current process flow to at least one existing process flow and which allocates joint resources of a computer system for said current process flow and said at least one existing process flow; and

(a2) defining said first event at least partially with said mergeFlow operation.

472. The development system as claimed in claim 471, wherein said mergeFlow operation assigns a unique flow index number to said current process flow.

473. The development system as claimed in claim 472, wherein said flow index number of said current process flow is different than a flow index number of said at least one existing process flow.

474. The development system as claimed in claim 428, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed when said match operation is a success, and

wherein said first event completes execution when said dedicated instruction is executed.

475. The development system as claimed in claim 474, wherein the dedicated instruction indicates a successful completion of said first event.

476. The development system as claimed in claim 428, wherein said match operation comprises a dedicated instruction,

wherein said dedicated instruction is executed when said match operation is unsuccessful, and

wherein said first event completes execution when said dedicated instruction is executed.

477. The development system as claimed in claim 476, wherein the dedicated instruction indicates an unsuccessful completion of said first event.

478. The development system as claimed in claim 475, wherein said first event generates successful completion data when said first event is successfully completed, and

wherein at least a second event utilizes said successful completion data to determine whether or not said first event has been successfully completed.

479. The development system as claimed in claim 477, wherein said first event generates unsuccessful completion data when said first event is not successfully completed, and

wherein at least a second event utilizes said unsuccessful completion data to determine whether or not said first event has been successfully completed.

480. The development system as claimed in claim 425, wherein said operation (a) comprises:

(a1) defining a variable as part of said first event; and

(a2) defining an alias for said variable as part of said first event, wherein said variable is capable of being referenced via said alias.

481. The development system as claimed in claim 480, further comprising:

(c) defining a second event, wherein a variable reference to said alias is part of said second event and wherein said second event can utilize said variable via said variable reference.

482. The development system as claimed in claim 480, wherein said variable has a plurality of subparts, and

wherein said alias is defined as a subset of said variable containing less than all of

said subparts such that said variable reference corresponds to only said subset.

483. The development system as claimed in claim 482, wherein said subparts are subfields of said variable.

484. The development system as claimed in claim 425,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable equals said at least said particular value.

485. The development system as claimed in claim 484, wherein said particular operation is a match operation.

486. The development system as claimed in claim 484, wherein said particular operation is an event definition.

487. The development system as claimed in claim 425,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable in a particular operation and setting said variable equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable does not equal said at least said particular value.

488. The development system as claimed in claim 487, wherein said particular operation is a match operation.

489. The development system as claimed in claim 487, wherein said particular operation is an event definition.

490. The development system as claimed in claim 425,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable is within said particular range.

491. The development system as claimed in claim 490, wherein said particular operation is a match operation.

492. The development system as claimed in claim 490, wherein said particular operation is an event definition.

493. The development system as claimed in claim 425,

wherein said operation (a) comprises:

(a1) defining a variable as at least part of said first event, and

wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable in a particular operation and setting said variable equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable is not within said particular range.

494. The development system as claimed in claim 493, wherein said particular operation is a match operation.

495. The development system as claimed in claim 493, wherein said particular operation is an event definition.

496. The development system as claimed in claim 481, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable reference equals said particular value.

497. The development system as claimed in claim 496, wherein said particular operation is a match operation.

498. The development system as claimed in claim 496, wherein said particular operation is an event definition.

499. The development system as claimed in claim 481, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular value by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular value in said particular operation; and

(e) determining that said particular operation is a success if said variable reference does not equal said particular value.

500. The development system as claimed in claim 499, wherein said particular operation is a match operation.

501. The development system as claimed in claim 499, wherein said particular operation is an event definition.

502. The development system as claimed in claim 481, wherein said alias is

defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable reference is within said particular range.

503. The development system as claimed in claim 502, wherein said particular operation is a match operation.

504. The development system as claimed in claim 502, wherein said particular operation is an event definition.

505. The development system as claimed in claim 481, wherein said alias is defined as a particular subfield of said variable and wherein said routine further comprises:

(d) constraining said variable to be restricted to at least a particular range of values by referring to said variable reference in a particular operation and setting said variable reference equal to said at least said particular range in said particular operation; and

(e) determining that said particular operation is a success if said variable reference is not within said particular range.

506. The development system as claimed in claim 505, wherein said particular operation is a match operation.

507. The development system as claimed in claim 505, wherein said particular operation is an event definition.

508. The development system as claimed in claim 425, wherein said first event is described in at least a first layer of said communication model.

509. The development system as claimed in claim 508, wherein said first layer is one of said application layer, said presentation layer, said session layer, said transport layer, and said network layer of said communication model.

510. The development system as claimed in claim 509, the routine further comprising:

(c) transforming said first event into actions in said first layer of said communication model.

511. The development system as claimed in claim 509, the routine further comprising:

(c) transforming said first event into actions in a second layer of said communication model, wherein said second layer is lower in said communication model than said first layer.

512. The development system as claimed in claim 425, wherein said operation

(a) comprises:

(a1) defining a first throw operation, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction; and

(a2) defining said first event at least partially with said first throw instruction, and

wherein said routine further comprises:

(d) defining a resynchronization operation.

513. The development system as claimed in claim 512, the routine further comprising:

(e) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed.

514. The development system as claimed in claim 513, wherein said resynchronization operation identifies said first exception name.

515. The development system as claimed in claim 513, the routine further comprising:

(f) defining a second throw operation, wherein said second throw operation identifies a second exception name corresponding to said second throw instruction, and

wherein said resynchronization operation identifies said first exception name and said second exception name, and

wherein said operation (e) comprises:

(e1) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said

first exception name, and

(e2) resynchronizing to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.

516. A development system for creating an application program, comprising:

a computer readable medium containing software having instructions; and

a compiler that compiles said instructions into network triggers for a network processor,

wherein said software program contains instructions to perform a routine, comprising:

(a) defining a matching operation that occurs in said network system, wherein said matching operation is defined via software and detects an occurrence corresponding to information transmitted over said network system and corresponding to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model, and

wherein said occurrence is identified in said matching operation; and

(b) controlling at least a portion of said network system based on said matching operation.

517. The development system as claimed in claim 516, wherein said operation (a) comprises:

(a1) defining a first event via said software; and

(a2) defining said first event at least partially with said matching operation.

518. The development system as claimed in claim 516, wherein said matching operation identifies whether or not a series of data within a process flow corresponds to said occurrence.

519. The development system as claimed in claim 516, wherein said occurrence corresponds to a predefined event, and

wherein said matching operation identifies whether or not said predefined event has occurred.

520. The development system as claimed in claim 516, the routine further comprising:

(c) repeatedly performing said matching operation within a process flow.

521. The development system as claimed in claim 520, the routine further comprising:

(d) discontinuing performance of said matching operation when said matching operation detects said occurrence.

522. The development system as claimed in claim 520, wherein said operation (c) comprises:

(c1) updating whether or not said matching operation is a success or a failure each time said matching operation is performed,

wherein said matching operation is said success when said matching operation detects said occurrence, and

wherein said matching operation is said failure when said matching operation does

not detect said occurrence.

523. The development system as claimed in claim 516, wherein said matching operation is performed on a plurality of process flows.

524. The development system as claimed in claim 516, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected after said specified location.

525. The development system as claimed in claim 516, wherein said matching operation defines a specified location within information transmitted on said network system and wherein said matching operation is successful only if said occurrence corresponding to said information is detected at said specified location.

526. The development system as claimed in claim 516, the routine further comprising:

(c) defining an otherwise operation, wherein said otherwise operation is performed when said matching operation is a failure.

527. The development system as claimed in claim 526, wherein said otherwise operation is performed immediately after said matching operation when said matching operation is said failure.

528. The development system as claimed in claim 516, wherein said occurrence

is a variable to be detected, and

wherein said matching operation is defined by a match instruction that comprises a var modifier that identifies said variable to be detected.

529. The development system as claimed in claim 528, wherein said matching instruction comprises a variable constraint, and

wherein said matching operation is satisfied when said variable has at least a certain value identified by said variable constraint.

530. The development system as claimed in claim 528, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said matching operation to determine whether or not said variable exists in only said current datagram.

531. The development system as claimed in claim 529, wherein said match instruction comprises an immediate modifier, and

wherein said immediate modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in only said current datagram.

532. The development system as claimed in claim 528, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said matching operation to determine whether or not said variable exists in at least one process flow identified by said flow modifier.

533. The development system as claimed in claim 529, wherein said match instruction comprises a flow modifier, and

wherein said flow modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in at least one process flow identified by said flow modifier.

534. The development system as claimed in claim 528, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said matching operation to determine whether or not said variable exists at a particular location identified by said at modifier.

535. The development system as claimed in claim 534, wherein said particular location comprises a particular layer of a communication model.

536. The development system as claimed in claim 534, wherein said particular location identifies at least one of (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

537. The development system as claimed in claim 529, wherein said match instruction comprises an at modifier, and

wherein said at modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value at a particular location identified

by said at modifier.

538. The development system as claimed in claim 537, wherein said particular location comprises a particular layer of a communication model.

539. The development system as claimed in claim 537, wherein said particular location identifies (1) a starting offset of an Internet protocol header in a datagram transmitted on said network system, (2) a starting offset of a payload in a datagram transmitted on said network system, (3) a starting offset of a transport header in a datagram transmitted on said network system, and (4) a starting offset of a first payload in a series of concatenated payloads in a process flow transmitted on said network system.

540. The development system as claimed in claim 528, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said matching operation to determine whether or not said variable exists at or after a particular location identified by said offset modifier.

541. The development system as claimed in claim 529, wherein said match instruction comprises an offset modifier, and

wherein said offset modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value at or after a particular location identified by said offset modifier.

542. The development system as claimed in claim 528, wherein said match

instruction comprises a direction modifier, and

wherein said direction modifier causes said matching operation to determine whether or not said variable exists in data travelling in a particular direction identified by said direction modifier.

543. The development system as claimed in claim 529, wherein said match instruction comprises a direction modifier, and

wherein said direction modifier causes said matching operation to determine whether or not said variable exists and has at least said certain value in data travelling in a particular direction identified by said direction modifier.

544. The development system as claimed in claim 516, wherein said occurrence is a predefined event to be detected, and

wherein said matching operation is defined by a match instruction that comprises an event modifier that identifies said predefined event to be detected.

545. The development system as claimed in claim 544,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable equals said at least said particular value.

546. The development system as claimed in claim 544,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular value by referring to said variable in said matching instruction and setting said variable equal to said at least said particular value, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable does not equal said at least said particular value.

547. The development system as claimed in claim 544,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said

variable falls within said at least said particular range.

548. The development system as claimed in claim 544,

wherein said routine further comprises:

(c) defining a variable as at least part of said predefined event,

wherein said operation (a) comprises:

(a1) defining a variable constraint in said matching instruction, wherein said variable constraint constrains said variable to be restricted to at least a particular range of values by referring to said variable in said matching instruction and setting said variable equal to said at least said particular range, and

wherein said operation (b) comprises:

(b1) determining that said matching operation is a success if said variable does not fall within said at least said particular range.

549. The development system as claimed in claim 516, wherein said matching operation is described in at least an application layer of a communication model.

550. The development system as claimed in claim 516, wherein said matching operation is described in at least a presentation layer of a communication model.

551. The development system as claimed in claim 516, wherein said matching operation is described in at least a session layer of a communication model.

552. The development system as claimed in claim 516, wherein said matching operation is described in at least a transport layer of a communication model.

553. The development system as claimed in claim 516, wherein said matching operation is described in at least a network layer of a communication model.

554. A development system for creating an application program, comprising:

a computer readable medium containing software having instructions; and

a compiler that compiles said instructions into network triggers for a network processor,

wherein said software program contains instructions to perform a routine, comprising:

(a) defining a concurrent operation, wherein said concurrent operation comprises a first group of operations comprising at least a first operation and a second operation that are concurrently performed and wherein said first group of operations corresponds to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said concurrent operation.

555. The development system as claimed in claim 554, wherein said concurrent operation ceases performing all unsuccessful operations of said first group of operations when one operation of said first group of operations is a success.

556. The development system as claimed in claim 555, the routine further comprising:

(c) defining a third operation that is performed if all of said first group of operations are not a success.

557. A development system for creating an application program, comprising:

a computer readable medium containing software having instructions; and

a compiler that compiles said instructions into network triggers for a network processor,

wherein said software program contains instructions to perform a routine, comprising:

(a) defining a first throw operation, wherein said first throw operation identifies resynchronization operation corresponding to said first throw operation, and

wherein said first throw operation causes a specific operation corresponding to said resynchronization operation to be performed and corresponding to information transmitted over said network system in at least one of application, presentation, session, transport, and network layers of a communication model; and

(b) controlling at least a portion of said network system based on said first throw operation and said resynchronization operation.

558. The development system as claimed in claim 557, wherein said first throw operation identifies a first exception name corresponding to said first throw instruction.

559. The development system as claimed in claim 558, wherein said operation (b) comprises:

(b1) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed.

560. The development system as claimed in claim 559, wherein said resynchronization operation identifies said first exception name.

561. The development system as claimed in claim 559, the routine further comprising:

(c) defining a second throw operation, wherein said second throw operation identifies a second exception name corresponding to said second throw instruction, and

wherein said resynchronization operation identifies said first exception name and said second exception name, and

wherein said operation (b1) comprises:

(b1a) resynchronizing to an instruction immediately following said resynchronization operation when said first throw operation is performed based on said first exception name, and

(b1b) resynchronizing to said instruction immediately following said resynchronization operation when said second throw operation is performed based on said second exception name.